8. Web Fundamentals

🌐 How the Web Works & Web Applications

1. How the Web Works

- The web is a **client-server model**:
  - The **client** (usually a browser) sends requests to a **web server** over HTTP/HTTPS.
  - The server processes the request (may involve databases, APIs, or other services) and sends back an HTTP response with HTML, CSS, JavaScript, or data.
- Communication uses **TCP/IP**, mostly on ports **80 (HTTP)** and **443 (HTTPS)**.
- URLs specify the resource location; DNS resolves domain names to IP addresses so clients can find servers.

2. Examples of Web Applications

- **Social media platforms** (Facebook, Twitter)
- **E-commerce websites** (Amazon, eBay)
- **Email clients** (Gmail, Outlook Web)
- **Content management systems** (WordPress, Joomla)
- **Online banking portals**
- **Online collaboration tools** (Google Docs, Slack)
- **Streaming services** (Netflix, YouTube)

3. Web 1.0 vs Web 2.0 vs Web 3.0

Feature	Web 1.0	Web 2.0	Web 3.0
Timeframe	Early 1990s – early 2000s	Mid 2000s – present	Emerging, mid-2020s onward
Content	Static pages	Dynamic, user-generated content	Decentralized, semantic web
User Interaction	Limited, read-only	Highly interactive, social	AI-driven, blockchain-based apps
Technology	HTML, basic CGI scripts	AJAX, APIs, social media, cloud	AI, ML, blockchain, decentralized networks
Data Control	Centralized	Centralized	Decentralized

4. PWA - Progressive Web Applications

- PWAs combine features of **websites and native apps**:
  - Can work offline or on poor networks (using caching, service workers)
  - Installable on devices without app stores
  - Responsive, fast, and reliable
- Offer near-native experience with push notifications, background sync, and hardware access.

---

5. How Does the Front-End Communicate with the Back-End?

- Front-end (browser, client-side code) communicates with the back-end via:
  - **HTTP requests** (GET, POST, PUT, DELETE, etc.) using REST APIs or GraphQL
  - **WebSockets** for real-time two-way communication
  - Data formats typically used: **JSON**, **XML**, or plain text
- The back-end processes requests, interacts with databases or services, and returns responses.

---

6. Stateful vs Stateless: What's the Difference?

Aspect	Stateful	Stateless
Definition	Server stores client session state	Server treats each request independently
Examples	Traditional web apps with sessions	RESTful APIs, HTTP is stateless
Complexity	Higher; needs session management	Simpler; easier to scale
Performance	Can be slower due to state maintenance	Faster; no session overhead

---

7. Structured vs Unstructured Data: What's the Difference?

Aspect	Structured Data	Unstructured Data
Format	Organized, easily searchable (tables)	Raw, unorganized (text, images, videos)
Examples	SQL databases, Excel sheets	Emails, social media posts, multimedia
Processing	Easy with traditional tools	Requires advanced processing (NLP, ML)
Storage	Relational databases	NoSQL, file systems, data lakes

---

8. Web Application Security Risks

- **Injection flaws** (SQL, command, LDAP injection)
- **Cross-Site Scripting (XSS)**

- **Broken Authentication and Session Management**
- **Sensitive Data Exposure**
- **XML External Entity (XXE) attacks**
- **Security Misconfiguration**
- **Broken Access Control**
- **Cross-Site Request Forgery (CSRF)**
- **Using Components with Known Vulnerabilities**
- **Insufficient Logging & Monitoring**

(Refer to **OWASP Top Ten** for full detail)

---

9. **Bug Bounty Programs**

- Programs where organizations invite security researchers to find and report vulnerabilities in their applications or infrastructure for rewards.
- Benefits:
  - Leverages community expertise to improve security
  - Encourages responsible disclosure
  - Helps find unknown vulnerabilities before attackers do
- Platforms: HackerOne, Bugcrowd, Synack
- Key to success: clear scope, rules, and communication channels.